

---

# Tipos Abstratos de Dados

---

Prof. Jussara Almeida

# Algoritmos e Estrutura de Dados

- **Algoritmo:**

- Sequência de ações executáveis para a solução de um determinado tipo de problema
- Exemplo: “Receita de Bolo”

- Em geral, algoritmos trabalham sobre **Estruturas de Dados**

- Conjunto de dados que representa uma situação real
- Abstração da realidade

- Estruturas de Dados e Algoritmos estão intimamente ligados

# Representação dos dados

- Dados podem estar representados (estruturados) de diferentes maneiras
- Normalmente, a escolha da representação é determinada pelas operações que serão utilizadas sobre eles
- Exemplo: números inteiros
  - Representação por palitinhos:  $II + IIII = IIIII$ 
    - Boa para pequenos números (operação simples)
  - Representação decimal:  $1278 + 321 = 1599$ 
    - Boa para números maiores (operação complexa)

# Programas

- Um programa é uma formulação concreta de um algoritmo abstrato, baseado em representações de dados específicas
- Os programas são feitos em alguma linguagem que pode ser entendida e seguida pelo computador
  - Linguagem de máquina
  - Linguagem de alto nível (uso de compilador)

# Tipos Abstratos de Dados (TADs)

- Agrupa a estrutura de dados juntamente com as operações que podem ser feitas sobre esses dados
- O TAD encapsula a estrutura de dados. Os usuários do TAD só tem acesso a algumas operações disponibilizadas sobre esses dados
- Usuário do TAD x Programador do TAD
  - Usuário só “enxerga” a interface, não a implementação

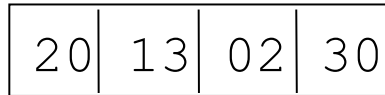
# Tipos Abstratos de Dados (TADs)

- Dessa forma, o usuário pode abstrair da implementação específica.
- Qualquer modificação nessa implementação fica restrita ao TAD
- A escolha de uma representação específica é fortemente influenciada pelas operações a serem executadas

# Exemplo: Lista de números inteiros

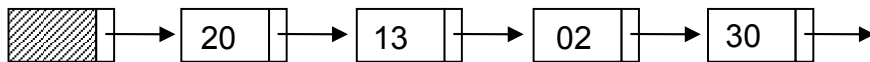
- Operações
  - Faz Lista Vazia
  - Insere número no começo da lista
  - Remove de uma posição i

Implementação por Vetores:



```
void Insere(int x, Lista L) {  
    for(i=0;...) {...}  
    L[0] = x;  
}
```

Implementação por Listas Encadeadas



```
void Insere(int x, Lista L) {  
    p = CriaNovaCelula(x);  
    L^.primeiro = p;  
    ...  
}
```

Programa usuário do TAD:

```
int main() {  
    Lista L;  
    int x;  
  
    x = 20;  
    FazListaVazia(L);  
    Insere(x, L);  
    ...  
}
```

# Implementação de TADs

- Em linguagens orientadas por objeto (C++, Java) a implementação é feita através de classes
- Em linguagens estruturadas (C, pascal) a implementação é feita pela definição de tipos juntamente com a implementação de funções
- Nesta disciplina, vamos utilizar os conceitos de linguagens estruturadas
  - C (**typedef e structs**) e
  - Pascal (**type, record**)
- Orientação por objetos (classes, etc) vai ser vista em disciplinas posteriores (POO, Prog. Modular)

# Estruturas

- Uma estrutura é uma coleção de uma ou mais variáveis, possivelmente de tipos diferentes, colocadas juntas sob um único nome para manipulação conveniente
- Por exemplo, para representar um aluno são necessárias as informações nome, matrícula, conceito
- Ao invés de criar três variáveis, é possível criar uma única variável contendo três campos.
- Em C: struct
- Em Pascal: record

# Estruturas (structs) em C

```
struct Aluno {
    char nome[100];
    int matricula;
    char conceito;
};

main() {
    struct Aluno al, aux;

    al.nome = "Luiz"
    al.matricula = 200712;
    al.conceito = 'A';
    aux = al;
    printf("%s", aux.nome);
}
```

**al:**

Luiz	
200712	A

**aux:**

Luiz	
200712	A

# Declaração de Tipos

- Para simplificar, uma estrutura ou mesmo outros tipos de dados podem ser definidos como um novo tipo
- Uso da construção **typedef**

```
typedef struct {  
    char nome[100];  
    int matricula;  
    char conceito;  
} TipoAluno;  
  
typedef int[10] Vetor;
```

```
int main() {  
    TipoAluno al;  
    Vetor v;  
  
    ...  
}
```

# TADs

- Para implementar um Tipo Abstrato de Dados, usa-se a definição de tipos juntamente com a implementação de funções que agem sobre aquele tipo
- Como boa regra de programação, evita-se acessar o dado diretamente, fazendo o acesso só através das funções
  - Note que não há uma forma de proibir o acesso.

# TADs em C

- Uma boa técnica de programação é implementar os TADs em arquivos separados do programa principal
- Para isso geralmente separa-se a declaração e a implementação do TAD em dois arquivos:
  - NomeDoTAD.h : com a declaração
  - NomeDoTAD.c : com a implementação
- O programa ou outros TADs que utilizam o seu TAD devem dar um `#include` no arquivo `.h`

# Exemplo

- Implemente um TAD ContaBancaria, com os campos número e saldo onde os clientes podem fazer as seguintes operações:
  - Iniciar uma conta com um número e saldo inicial
  - Depositar um valor
  - Sacar um valor
  - Imprimir o saldo
- Faça um pequeno programa para testar o seu TAD

---

# ContaBancaria.h

```
// definição do tipo
typedef struct {
    int numero;
    double saldo;
} ContaBancaria;

// cabeçalho das funções
void Inicializa(ContaBancaria* conta, int numero, double saldo);
void Deposito (ContaBancaria* conta, double valor);
void Saque (ContaBancaria* conta, double valor);
void Imprime (ContaBancaria conta);
```

# ContaBancaria.c

```
#include <stdio.h>
#include "Contabancaria.h"

void Inicializa(ContaBancaria* conta, int numero, double saldo)
{
    (*conta).numero = numero;
    (*conta).saldo = saldo;
}

void Deposito (ContaBancaria* conta, double valor)
{
    (*conta).saldo += valor;
}

void Saque (ContaBancaria* conta, double valor)
{
    (*conta).saldo -= valor;
}

void Imprime (ContaBancaria conta)
{
    printf("Numero: %d\n", conta.numero);
    printf("Saldo: %f\n", conta.saldo);
}
```

# Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "ContaBancaria.h"

int main (int argc, char **argv)
{
    ContaBancaria conta1;
    Inicializa(&conta1, 918556, 300.00);
    printf("\nAntes da movimentacao:\n ");
    Imprime(conta1);
    Deposito(&conta1, 50.00);
    Saque(&conta1, 70.00);
    printf("\nDepois da movimentacao:\n ");
    Imprime (conta1);
}
```

# Exercício (foco nos algoritmos)

- Implemente um TAD **Número Complexo**
  - cada número possui os campos real e imaginário
  - Implemente as operações:
    - Atribui: atribui valores para os campos
    - Imprime: imprime o número da forma “ $R + Ci$ ”
    - Copia: Copia o valor de um número para outro
    - Soma: Soma dois números complexos
    - EhReal: testa se um número é real
- Faça uma pequena aplicação para testar o seu TAD.